REMARKS

Claims 1-7, 9-11, and 13-23 are pending. Claim 1 has been amended.

The Examiner has rejected claims 7 and 9-10 under 35 U.S.C. § 102(e) in view of Hogle et al. (U.S. Patent No. 6,560,626); and claims 1-6, 11, and 13-23 under 35 U.S.C. § 103(a) over Hogle et al. in view of Jones et al. (U.S. Patent No. 6,584,489).

Applicant's technology is fundamentally different than the cited prior art. In particular, the cited prior art is a single instruction stream processor, while Applicant's technology uses a multiple instruction stream processor. Applicant's technology provides a technique for given a stream back to a task when an operating system call blocks.

It is common in the prior art for an operating system to create the illusion that multiple tasks are running simultaneously by dividing up the processor's time among multiple tasks and allowing each task to execute instructions for a small amount of time, called a time slice. The processor cycles through these tasks rapidly enough that it appears to the user that all of the tasks are running simultaneously. The prior art cited by the Examiner exhibits these characteristics. For example, Hogle states the following:

> A computer's central processing unit (CPU) can only execute one instruction at a time. However, fast CPUs are capable of executing many instructions each second so that by switching between instructions in one program and those in another, the computer appears to the user to be running multiple programs simultaneously. Hogle, 1:12-17.

Jones asserts similar limitations:

> A typical conventional scheduling strategy is to divide the processor time resource into "time slices" having a uniform length. At the expiration of a time slice, a conventional scheduler identifies a thread to execute during the next time slice. Jones, 1:55-59.

In contrast, Applicant's technology is in fact capable of executing multiple threads simultaneously. Applicant's technology uses a multithreaded architecture (MTA) processor that has 128 streams that can each be simultaneously executing a thread. Specification, ¶4. A stream can be thought of as a virtual processor, and the cited prior art can be thought of as having only one stream that must be shared by all running tasks. In contrast, the processor used by Applicant's technology is able to simultaneously execute many tasks, each using a different stream or set of streams. This gives rise to various new scheduling techniques that are the subject of Applicant's claims.

One such technique involves allowing a running task to reassign a stream to be used for productive work when a thread makes a blocking operating system call. In the prior art systems, such a call would cause the scheduler to allow a thread from another task to run. Where there are enough streams for each task to run simultaneously in an MTA, it is desirable to give the stream back to the task with the blocked thread so that task can run other threads. There are two key differences between this system and the prior art. First, other tasks continue to run using their own streams regardless of what is happening with another task's streams. Second, the task whose thread is blocked gets the benefit of the stream that has been freed for other work by the blocking operating system call. In prior art systems, it would be necessary for the operating system to interrupt the blocked thread in order for any other task to run, and the processor would then be assigned to run the next ready thread which would likely be part of another task.

Hogle is directed at terminating a blocked thread without using excessive resources. The technique described by Hogle involves two threads, a first alertable thread that is blocked, and a second interrupting thread that wants to terminate the blocked thread. The second thread terminates the blocked thread by queuing an asynchronous procedure call (APC) against the blocked thread, which causes the operating system to wake up the blocked thread and throw an exception in the blocked thread's context that can be caught by user program code.

Applicant's technology does not relate to terminating blocked threads, but rather to allowing a processor stream to be used by another thread while a thread is blocked. Using Applicant's technique, another thread will run on the blocked thread's stream, and later when the operating system call completes, the blocked thread will be assigned a stream on which to finish executing.

Jones is directed at various techniques for scheduling a processor resource that can only execute one instruction at a time. As noted above, Applicant's technology executes many instructions at one time and is not directed at the kind of scheduling techniques necessary in the type of system described by Jones. In Applicant's technology when a stream is available to execute other instructions, the task is asked via a callback function which thread in that task should next be allowed to use the processor, rather than the operating system deciding by a scheduling process like that described in Jones.

Neither Hogle nor Jones, either alone or in combination, anticipate or render Applicant's technology obvious. Applicant respectfully requests that the above-noted rejections be withdrawn and the pending application be allowed.

In view of the above amendment, applicants believe the pending application is in condition for allowance.

Applicants believe no fee is due with this response. However, if a fee is due, please charge our Deposit Account No. 50-0665, under Order No. 324758001US3 from which the undersigned is authorized to draw.

Dated: June 24, 2005

Respectfully submitted,

By Maurice J. Pirio

Maurice J. Pirio
Registration No.: 33,273
PERKINS COIE LLP
P.O. Box 1247
Seattle, Washington 98111-1247
(206) 359-8000
(206) 359-7198 (Fax)
Attorney for Applicants